

Lab: Setting up GitHub the Environment



Estimated time needed: 15 minutes

What you will learn

In this lab, you will set up an online repository for the project and learn how to create folders and files in the **Skills Network Environment**. You will also understand how to view the output of your code.

Learning objectives

After completing this lab, you will be able to:

- Create an online repository.
- Create folders and files in **Skills Network Environment**.
- Create and run the react application.
- Perform Git operations.

Prerequisites

- You must have completed the prerequisite courses, especially the **Getting Started with Git and GitHub** course.
- You must have a GitHub account. If you want to set up a GitHub account, click [here](#) for the detailed steps.

Need to perform Git commands

It is crucial to follow a few essential steps to ensure the proper management and persistence of your data in a GitHub repository:

- **Regular updates:** Whenever you make changes or add new components to your project, adding, committing, and pushing the updates to your GitHub repository is essential. This process ensures that your latest work is safely stored and accessible to collaborators.
- **Session persistence:** During an active session, your data remains accessible. However, it's important to note that if your session expires or you log out, you will need to clone the repository again to resume work.

Task 1: Create your repository and generate a personal access token

1. Create a blank public GitHub repository in your GitHub account without creating any README.md file for this. Make sure you set your repository to public and name it accordingly.
2. You can find more information in the [GitHub Sign Up and Create Repo](#) lab.
3. To push files to a GitHub repository, you must have a **Personal Access Token** to make sure your authentication is related to your GitHub account.
4. To create a **Personal Access Token**, go to your GitHub account and click on your profile icon located in the top-right corner. Then click on **Settings**.



javaScriptEssential

Your codespaces

Your organizations

Your enterprises

Your stars

Your sponsors

Your gists

Upgrade

Try Enterprise

Try Copilot

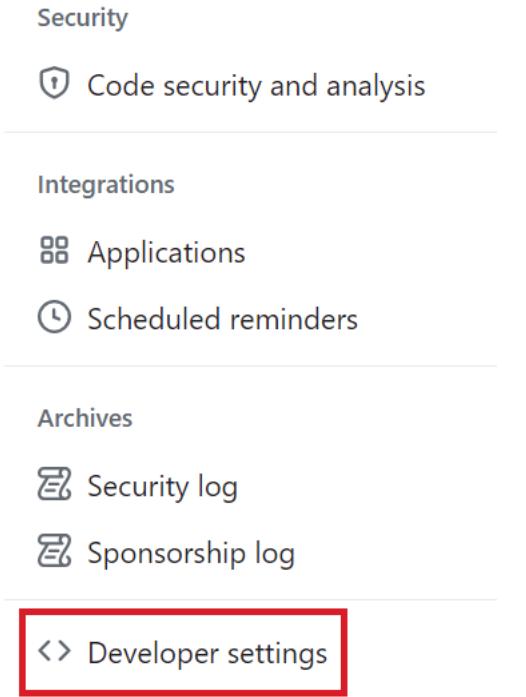
Feature preview

Settings

GitHub Docs

GitHub Support

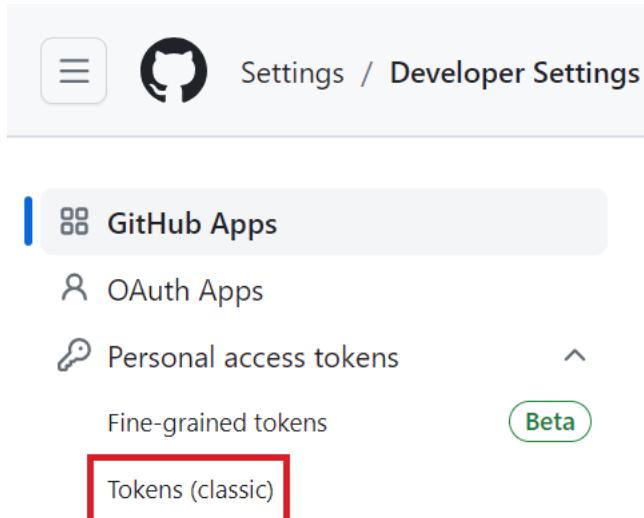
5. Next, select **Developer settings**. This option is typically available towards the bottom of the window.



The screenshot shows the GitHub Developer settings menu. It includes sections for Security, Integrations, Archives, and a highlighted 'Developer settings' option, which is enclosed in a red box.

- Security
- Code security and analysis
- Integrations
- Applications
- Scheduled reminders
- Archives
- Security log
- Sponsorship log
- <> **Developer settings**

6. Navigate to **Tokens (classic)** under **Personal access tokens**.



The screenshot shows the 'Developer Settings' page with the 'Personal access tokens' section selected. The 'Tokens (classic)' option is highlighted with a red box.

- GitHub Apps
- OAuth Apps
- Personal access tokens
 - Fine-grained tokens Beta
 - Tokens (classic)**

7. To generate an access token, click **Generate a personal access token**.

Personal access tokens (classic)

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the API.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of OAuth tokens for making requests over HTTPS, or can be used to authenticate to the API over Basic Authentication.

8. In the **Generate token** page, fill in the required details and click the **repo** checkbox to enable access for git commands.

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used as a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Auth](#).

Note

javaScriptEssential

What's this token for?

Expiration *

30 days



The token will expire on Tue, Sep 5 2023

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

repo

Full control of private repositories

repo:status

Access commit status

repo_deployment

Access deployment status

public_repo

Access public repositories

repo:invite

Access repository invitations

security_events

Read and write security events

9. Then, click **Generate token**.

admin:ssh_signing_key

Full control of public user SSH signing keys

write:ssh_signing_key

Write public user SSH signing keys

read:ssh_signing_key

Read public user SSH signing keys

Generate token

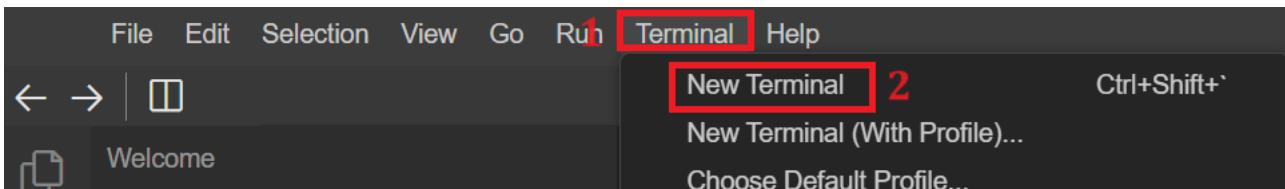
Cancel

10. Your personal access token will be generated. The token is only valid for **30 days**. You will need to generate a new token once the current token expires.

REMEMBER: Make sure to copy your personal access token now. You won't be able to see it again! In case you forgot to save it or misplaced it, delete the already created token and generate new one.

Task 2: Create files in Skills Network Environment

- If a terminal is not open in the Skills Network Environment, select the "Terminal" tab at the top-right of the window, then "New Terminal" from the dropdown. These are shown in the given screenshot.



2. Follow the given steps to create react application:

- Write the given command in the terminal:

```
npm create vite@latest
```

- The terminal should look similar to the given screenshot.

```
theia@theia-richaar:/home/project$ npm create vite@latest
```

- It will then ask if you are ok to proceed, for then you need to hit **Enter**.

```
theia@theia-richaar:/home/project$ npm create vite@latest
Need to install the following packages:
  create-vite@5.2.2
Ok to proceed? (y)
```

3. In next step, it will ask to enter project name as shown in the given screenshot.

```
? Project name: > vite-project
```

- Write project named `learning_react` and hit **Enter**. Make sure that project name should be in lowercase. You can see the given screenshot for reference.

```
? Project name: > learning_react
```

4. Select a framework from the list which will be displayed there. Use arrow keys to select **React** and hit **Enter**.

```
? Select a framework: > - Use arrow-keys. Return to exit.
  Vanilla
  Vue
> React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

5. Select the **JavaScript** variant using arrow keys and then hit **Enter**.

```
? Select a variant: >
  TypeScript
  TypeScript + SWC
> JavaScript
  JavaScript + SWC
  Remix ↵
```

- After completing above steps, it will create one folder with the application name `learning_react`. It will ask to run certain commands which will be displayed in terminal as shown in given screenshot.

```
Done. Now run:
```

```
cd learning_react
npm install
npm run dev
```

6. Now you need to go inside the application folder which you have created using terminal. For this, perform given command.

```
cd learning_react
```

7. Perform the `npm install` command by writing it in the terminal. It will install all required files to run the React application.

```
npm install
```

8. Now go to `package.json` under `learning_react` folder. Include given code in `preview` key under `script` object.

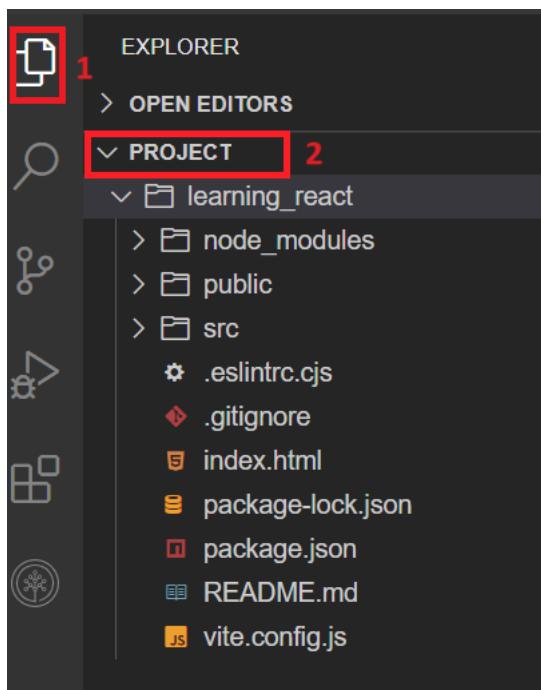
```
"preview": "vite build; vite preview --host"
```

- Make sure that the `script` object should look like given screenshot.

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings=0",
  "preview": "vite build; vite preview --host"
},
```

9. The next steps will allow you to run your React application.

Now click on the **Explorer** icon shown at number 1 in given screenshot, then expand the **project** folder by clicking on the twisty arrow. You will see the entire folder structure for your React application.



10. After this, enter the `npm run preview` command in the terminal to run the application.

```
npm run preview
```

Your terminal will display a line beginning with the word Local followed by the localhost path and a port number.

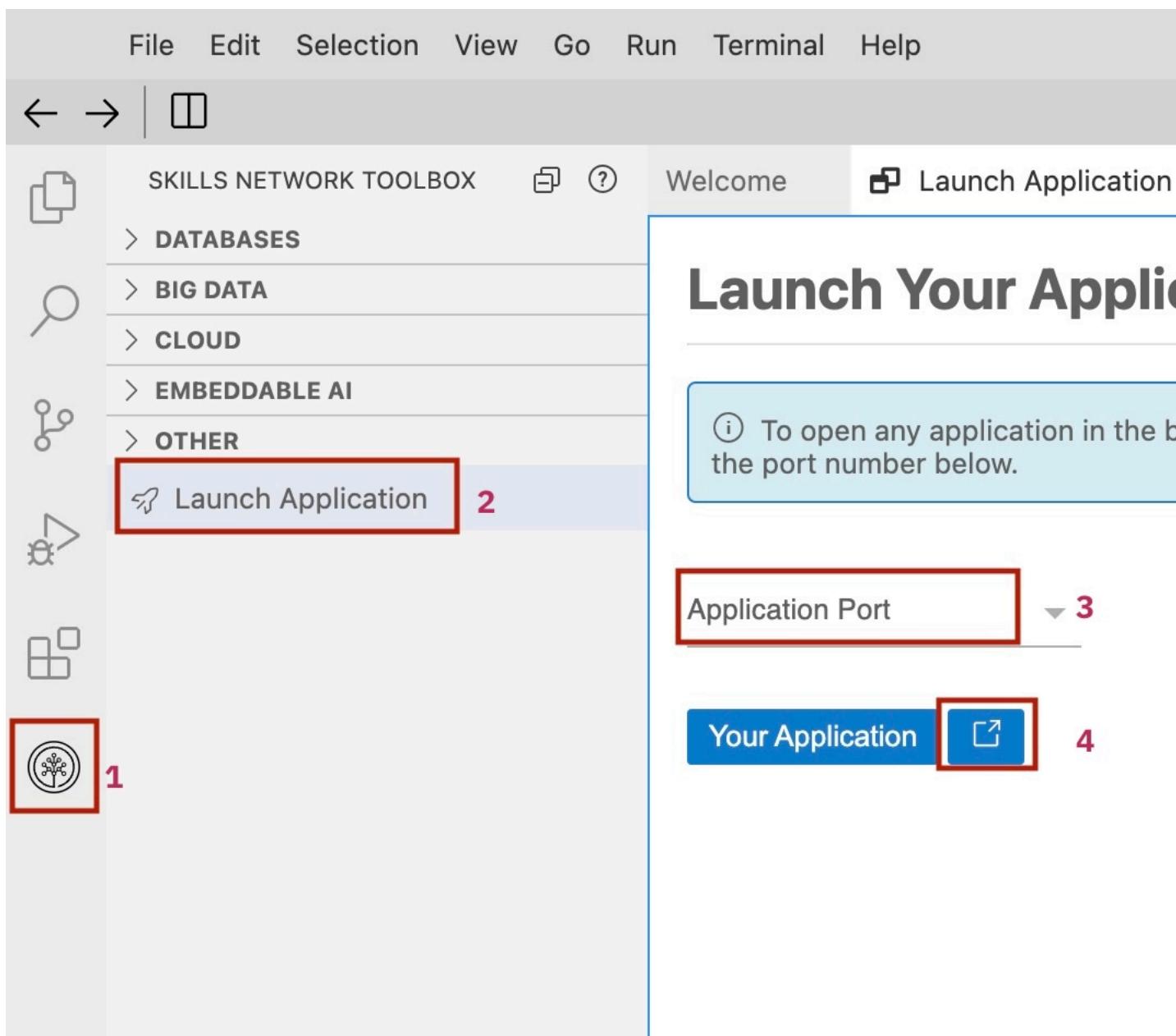
You can see a sample screen below, where the port number is 4173. This port will run your React application.

```
✓ built in 1.18s
→ Local: http://localhost:4173/
→ Network: http://172.22.175.156:4173/
→ press h + enter to show help
```

Task 3: Check your output

Next, you will take the required steps to view your output in a browser.

1. Select the **Skills Network** icon on the lower left of the IDE environment, shown at number one in the screenshot. This will open the "Skills Network Toolbox."
2. This action will open the **Skills Network Toolbox**. Next, click **Launch Application** (refer to number 2).
3. Enter port number **4173** in **Application Port** (refer to number 3) and click 



4. The React application will open your default browser, where you will see the output as given below.

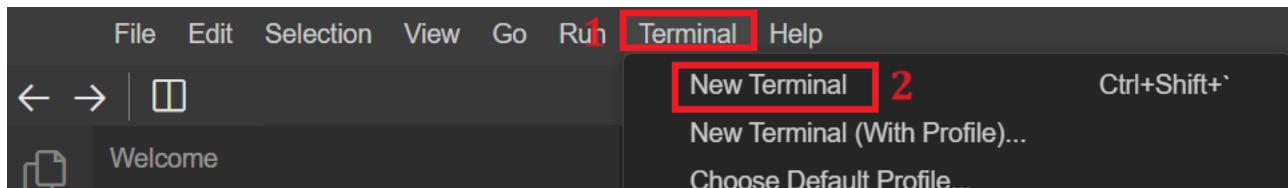


Vite + React

A screenshot of a web browser window. At the top, there's a header with a light gray background containing the text "count is 0". Below the header, the main content area displays the text "Edit src/App.jsx and save to test HMR". At the bottom of the content area, there's a link that says "Click on the Vite and React logos to learn more".

Task 4: Perform Git commands in the terminal

1. Now, select the "Terminal" tab at the top-right of the window, and then "New Terminal" from the dropdown, as indicated in the screenshot.



Make sure that your terminal path reads `/home/project/ cd <learning_react>` where you should replace `<learning_react>` with your own project name.

2. Next, initialize the terminal to create a Git repository so you can perform all Git commands in this terminal. Use the provided command and press **Enter**.

```
git init
```

3. Then you need to configure the terminal to perform necessary commands within the **Skills Network Environment**.

```
git config --global --add safe.directory /home/project
```

Press **Enter**

Important

The `git config` command will help you to work inside your project folder environment using git commands.

4. Next, you need to config git with your email address. Replace `you@example.com` with your email address. Be sure your email address is in double quotes.

```
git config --global user.email "you@example.com"
```

Press **Enter**.

Now you should use the `config` command to set your username. Replace *Your Name* with your name and be sure to use double quotes, as shown.

```
git config --global user.name "Your Name"
```

5. Next, perform `git add` and `git commit` to save the changes for GitHub repository. Run the following commands one after another:

```
git add --a
```

```
git commit -m "initial commit"
```

6. Perform `git push` commands to push the files into your GitHub repository. You need to set your main branch in your GitHub repository.

```
git branch -M main
```

7. Add your GitHub repository URL in an origin variable. Also replace `<git-repo-url>` with your GitHub repository URL.

For example:

```
git remote add origin https://github.com/<youraccountname>/<yourrepositoryname>
```

```
git remote add origin <git-repo-url>
```

Press **Enter**

8. Then, push the content of your file to your GitHub repository:

```
git push -u origin main
```

Press **Enter**

Important

*The above command sets your first push command to upload **all** of your changes directly to the main branch. You only need to perform this once. After this git push will also do the same.*

9. While pushing the files in GitHub using `git push` command, it will ask you to enter your username and password for your GitHub account in the terminal.

- Enter your username and then press enter.
- For your password, you need to paste your **Personal Access Token** generated in step 1.

Important

Upon pasting your **Personal Access Token** into the terminal, it won't display for security reasons, but it's there. Simply hit enter, and it will push your files and folders to the GitHub repository.

10. All your files have been pushed to your GitHub directory.

Note: You can also refer to [Hands-on Lab: Getting Started with Branches using Git Commands](#) for more details on git commands.

Task 5: Deploy using GitHub Pages

1. To deploy your react application in GitHub you need to install `gh-pages`. This allows you to use it as a tool for deploying your project to GitHub Pages. Perform given command in the terminal

```
npm install gh-pages --save-dev
```

2. Add given lines before "build": "vite build" in package.json file.

```
"predeploy": "npm run build",
"deploy": "gh-pages -d dist",
```

```
"scripts": {
  "dev": "vite",
  "predeploy": "npm run build",
  "deploy": "gh-pages -d dist",
  "build": "vite build",
  "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warn
```

3. Then in the vite.config.js file add this line before plugins: [react()]

```
base: "/YOUR_REPOSITORY_NAME",
```

```
export default defineConfig({
  base: "/YOUR_REPOSITORY_NAME",
  plugins: [react()],
})
```

Note: Instead of <YOUR_REPOSITORY_NAME> write your own repository name such as assume if your github repository name is learning_react the it should look like base: "/Learning_react"

- Now perform deploy command in the terminal to executes the "deploy" script defined in the package.json file, deploying the project to GitHub Pages using the gh-pages tool.

```
npm run deploy
```

```
theia@theia-ritikaj:/home/project/react$ npm run deploy
> react@0.0.0 predeploy
> npm run build

> react@0.0.0 build
> vite build

vite v5.3.1 building for production...
✓ 34 modules transformed.
dist/index.html          0.49 kB | gzip:  0.30 kB
dist/assets/react-CHdo91hT.svg 4.13 kB | gzip:  2.14 kB
dist/assets/index-DiwrgTda.css 1.39 kB | gzip:  0.72 kB
dist/assets/index-DQ9y_QkT.js 143.38 kB | gzip: 46.06 kB
✓ built in 1.20s

> react@0.0.0 deploy
> gh-pages -d dist

Username for 'https://github.com': [REDACTED]
Password for 'https://[REDACTED]@github.com':
Published
theia@theia-ritikaj:/home/project/react$
```

Note: Whenever you make any change to your code you need to save all your files and perform git commands for them.

- Perform git add and git commit commands to update changes in your code. Then perform git push command to update your GitHub repository for proper code management.
- Go to your GitHub repository. Then, navigate to your site's repository that you created.
- Under your repository name, click **Settings**.



- Navigate to the left hand side navigation bar. In the **Code and Automation** section of the sidebar, click **Pages**.

General

Access

Collaborators

Moderation options ▾

Code and automation

Branches

Tags

Rules ▾

Actions ▾

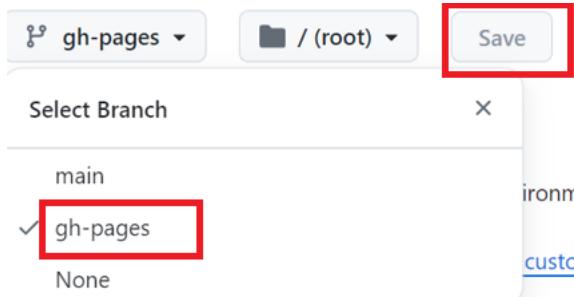
Webhooks

Environments

Codespaces

Pages

9. You will see the page shown below. Click the drop down menu where you see **None**, then click **gh-pages**, and then click the **Save** button.



10. Refresh your page again, and you will see the link, just as below. Instead of **shoppingreact**, you will see your github repository name.

Your site is live at [https://\[REDACTED\].github.io/shoppingreact/](https://[REDACTED].github.io/shoppingreact/)

Note: If you are not able to see the link, please wait for (1-2) minutes and refresh the page again.

11. Click above generated link to see your live website.



Vite + React

count is 0

Edit `src/App.jsx` and save to test HMR

Click on the Vite and React logos to learn more

Note:

1. If you haven't made any changes to your default React application, you should see a similar output for task 3, step 6, as shown below.
2. After deploying on GitHub Pages, it may take some time for all contents and images to appear properly. Please wait a few extra minutes for the application to load completely.

By adhering to these guidelines, you can maintain a well-organized and efficient GitHub repository, ensuring your work is securely stored and easily accessible to you and your collaborators.

Author(s)

Richa Arora

© IBM Corporation. All rights reserved.